R.B.F. and Batch Mode Training

Limitations:

3 inputs; 2 hidden; 2 output

Given Training data

X1	X2	X3	Y1	Y2
6	-11	0.1	-0.5	25
8	0	1	8	-50
10	11	10	24	210
12	22	100	-2	0

Scaling:

Input range given: {-1, 1} Output range {0, 1}

Input range implemented: {0.001, 1}

Scaling Functions:

Linear scaling: $\mathbf{X} = \mathbf{X}_{\mathbf{min}} + (x - x_{min})/(x_{max} - x_{min}) * (\mathbf{X}_{\mathbf{max}} - \mathbf{X}_{\mathbf{min}})$

Linear descaling: $x = x_{min} + (\mathbf{X} - \mathbf{X}_{\mathbf{min}})/(\mathbf{X}_{\mathbf{max}} - \mathbf{X}_{\mathbf{min}}) * (x_{max} - x_{min})$

Log scaling: $\mathbf{X} = \mathit{ln}(x - x_{min})$

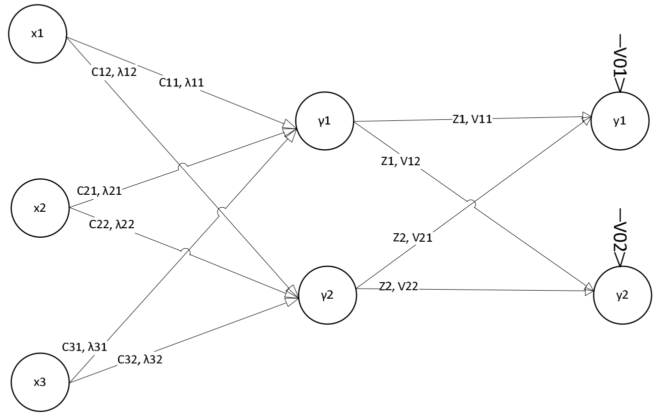
Log descaling: $x = x_{min} + \exp(\mathbf{X})$

Linear Scaled Training Data

Output		Implemented Input
Ymin = 0	ymin = -50	Xmin = 0.001; x1min = 6 ; x2min = -11; x3min = 0.1
Ymax = 1	ymax = 210	Xmax = 1 ; x1max = 12; x2max = 22; x3max = 100

Given input range			Implemented Input range			Expected output	
X1	X2	X3	X1	X2	X3	Y1	Y2
-1	-1	-1	0.001	0.001	0.001	0.1904	0.2885
-0.333	-0.333	-0.9820	0.334	0.334	0.01	0.2231	0.0000
0.333	0.333	-0.8018	0.667	0.667	0.1	0.2846	1.0000
1	1	1	1	1	1	0.1846	0.1923

Visual Neural Network



Neural Networks Homework 2 Edris Amin

R.B.F. and Batch Mode Training

Feed forward equations

Input: [x1; x2; x3]

$$C_{ij} = \begin{array}{c} in\backslash out \\ x1 \\ x2 \\ x3 \end{array} \begin{array}{c} \begin{array}{cc} h1 & h2 \end{array} \\ \begin{bmatrix} c11 & c12 \\ c21 & c22 \\ c31 & c32 \end{bmatrix} \end{array} \qquad \lambda_{ij} = \begin{array}{c} in\backslash out \\ x1 \\ x2 \\ x3 \end{array} \begin{array}{c} \begin{array}{cc} h1 & h2 \end{array} \\ \begin{bmatrix} \lambda 11 & \lambda 12 \\ \lambda 21 & \lambda 22 \\ \lambda 31 & \lambda 32 \end{bmatrix} \end{array} \qquad \gamma_j = \sqrt{\Sigma_{i=1}^{n=2}\left(\frac{x_i - c_{ij}}{\lambda_{ij}}\right)^2}$$

$$Z_j = \begin{array}{c} in\backslash out \\ h1 \\ h2 \\ h3 \end{array} \begin{array}{c} \begin{array}{cc} y1 & y2 \end{array} \\ \begin{bmatrix} Z1 & Z1 \\ Z2 & Z2 \\ Z3 & Z3 \end{bmatrix} \end{array} \qquad V_{jk} = \begin{array}{c} in\backslash out \\ h1 \\ h2 \\ h3 \\ bias \end{array} \begin{array}{c} \begin{array}{cc} y1 & y2 \end{array} \\ \begin{bmatrix} V11 & V12 \\ V21 & V22 \\ V31 & V32 \\ V01 & V02 \end{bmatrix} \end{array} \qquad y_k = V_{0k} + \sum_{j=1}^{N=2} V_{jk} * Z_j$$

$$Z_j = \exp\left(-{\gamma_j}^2\right)$$

Error equation

$E_k = ½*(y_k - d_k)^2$; k = 1, 2

Backpropagation Derivitives and update equations

$$\frac{dE_k}{dy_k} = (y_k - d_k); \quad \frac{dy_1}{dZ_1} = V_{11}; \quad \frac{dy_1}{dZ_2} = V_{21}; \quad \frac{dy_2}{dZ_1} = V_{12}; \quad \frac{dy_2}{dZ_2} = V_{22}; \quad \frac{dy_k}{dV_{0k}} = 0$$

$$\frac{dy_k}{dV_{1k}} = Z_1; \quad \frac{dy_k}{dV_{2k}} = Z_2; \quad \frac{dZ_j}{d\gamma_j} = -2\gamma_j * Z_j; \quad \frac{d\gamma_j}{dc_{ij}} = \frac{(c_{ij} - x_i)}{\lambda_{ij}^2 * \gamma_j};$$

$$\frac{d\gamma_j}{d\lambda_{ij}} = -\frac{(c_{ij} - x_i)^2}{\lambda_{ij}^3 * \gamma_j}; \quad \frac{dE_k}{dV_{jk}} = \frac{dE_k}{dy_k}\frac{dy_k}{dV_{jk}} = (y_k - d_k) * Z_j$$

$$\frac{dE_k}{dc_{ij}} = \frac{dE_k}{dy_k}\frac{dy_k}{dZ_j}\frac{dZ_j}{d\gamma_j}\frac{d\gamma_j}{dc_{ij}} = (y_k - d_k) * V_{kj} * -2\gamma_j * Z_j * \left(\frac{(c_{ij} - x_i)}{\lambda_{ij}^2 * \gamma_j}\right);$$

$$\frac{dE_k}{d\lambda_{ij}} = \frac{dE_k}{dy_k}\frac{dy_k}{dZ_j}\frac{dZ_j}{d\gamma_j}\frac{d\gamma_j}{d\gamma_{ij}} = (y_k - d_k) * V_{kj} * -2\gamma_j * Z_j * \left(-\frac{(c_{ij} - x_i)^2}{\lambda_{ij}^3 * \gamma_j}\right)$$

$$C_{new} = C_{old} - \eta\left(\Sigma_1^N\left(\frac{\Sigma_1^k \frac{dE_k}{dc_{ij}}}{k}\right)\right); \quad \lambda_{new} = \lambda_{old} - \eta\left(\Sigma_1^N\left(\frac{\Sigma_1^k \frac{dE_k}{d\lambda_{ij}}}{k}\right)\right)$$

$V_{new} = V_{old} - -\eta\left(\Sigma_1^N\left(\frac{dE_k}{dV_{jk}}\right)\right)$ N = number of samples; k = number of outputs

Results

The given scale for the inputs (-1, 1) was insufficient and would only converge for y values in samples 3 and 4. The output below shows the de-scaled y output (yo) and the expected values (dd):

yo =

-3.3078	-3.3078	23.9999	-2.0000
-1.0030	-1.0026	209.9989	-0.0001

dd =

-0.5000	8.0000	24.0000	-2.0000
25.0000	-50.0000	210.0000	0

Weights:

c =

0.9447	0.4419
0.2040	0.4153
0.7669	-0.3548

cinitial =

0.9330	0.9311
0.2017	0.0565
0.7592	0.2798

λ =

0.0678	0.2308
0.5118	0.5105
0.1580	1.2575

λinitial =

0.0562	0.4219
0.5083	0.6742
0.1461	0.7801

V =

0.9630	0.6885
0.1526	1.1791

Vinitial =

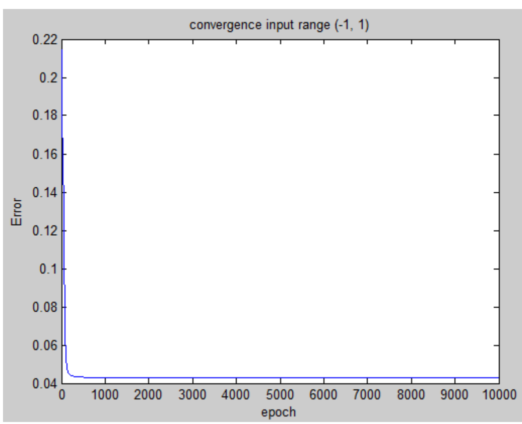
0.9628	0.6882
0.4984	0.0990

V0 =

0.1796
0.1885

V0initial =

0.1000
0.1000



Ranging the inputs between (0.001, 1) converged for all outputs except for sample 1 producing the following output:

yo =

-2.0378	7.9999	24.0000	-2.0000
-0.3056	-49.9993	210.0000	-0.0000

dd =

-0.5000	8.0000	24.0000	-2.0000
25.0000	-50.0000	210.0000	0

Weights:

c =

0.7129	0.7649
0.7586	0.3755
0.2060	0.1258

c initial =

0.8338	0.8316
0.8304	0.2598
0.2987	0.1867

λ =

0.1362	0.5580
0.4110	0.1083
0.5689	0.3769

λ initial =

0.1552	0.5391
0.4641	0.3540
0.5233	0.3507

V =

0.1220	0.9863
0.0891	-0.4417

V initial =

0.1897	0.8357
0.3842	0.0095

V0 =

0.1845
0.1911

V0 initial =

0.1000
0.1000

The same range (0.001, 1) sometimes wouldn't converge for sample 2:

yo =

-0.4713	-2.0000	24.0000	-2.0000
25.2888	0.0001	209.9999	-0.0000

dd =

-0.5000	8.0000	24.0000	-2.0000
25.0000	-50.0000	210.0000	0

Weights:

c =

0.7975	0.0905
0.6732	0.0755
0.1129	0.5417

cinitial =

0.9531	0.0933
0.3496	0.1900
0.3410	0.5070

λ =

0.8510	0.9684
-0.0837	0.0688
0.4579	0.4724

λinitial =

0.8149	0.9693
0.0216	0.1646
0.3844	0.5082

V =

0.1030	0.8322
0.0692	1.1446

Vinitial =

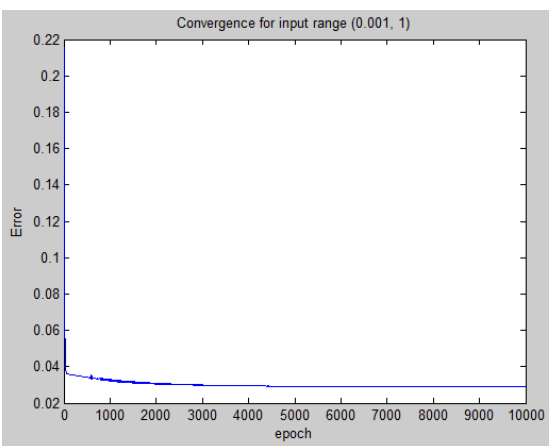
0.0515	0.3670
0.5848	0.9948

V0 =

0.1846
0.1923

V0initial =

0.1000
0.1000



Conclusion

With the given number of 2 hidden neurons the NN usually converged after about 200 – 400 epochs, with one of the outputs still being significantly off from the expected. After some experimentation with this NN I discovered that the NN was particularly sensitive to the number of hidden neurons. Incrementing the number of hidden neurons I found that the optimum was 20 hidden neurons. This case produced the correct output and showed the same convergence behavior.

Sample output with 20 Hidden Neurons:

```
yo =
  -0.5000    8.0000   24.0000   -2.0000
  25.0000  -50.0000  210.0000   -0.0000


c =
 Columns 1 through 5
   0.6351   -0.0084    0.5252    0.7501    0.8551
   0.4874    0.1066    0.4623    0.7552    0.5496
   0.8960    0.5353    0.4514    0.3478    0.5220

 Columns 6 through 10
   0.4354    0.3231    0.4549    0.0271    0.0182
   0.2517    0.8510    0.9044    0.5978    0.1155
   0.6633    0.1291    0.6038    0.2957    0.9871

 Columns 11 through 15
   0.3405    0.2740    0.4974    0.6737    0.4390
  -0.0643    0.9092    0.4032    0.1214    0.2168
   1.1454    0.9753    0.8465    0.8432    0.5015

 Columns 16 through 20
   0.3685    0.3112    0.1004    0.8328    1.1659
  -0.0140    0.6675    0.0719    0.0823    1.0182
   1.0575    0.1213    0.8690    0.9267    0.5390


λ =
 Columns 1 through 5
   0.1168    0.1575    0.8431    0.0196    0.0944
   0.2609    0.3827    0.8080    0.9118    0.9155
   0.7170    0.8222    0.8976    0.0713    0.0520

 Columns 6 through 10
   0.6868    0.9276    0.6506    0.8508    0.6283
   0.8913    0.2538    0.1107    0.5193    0.6198
  -0.1208    0.8140    0.0164    0.6083    0.6553

 Columns 11 through 15
   0.4301    0.4289    0.0358    0.7461    0.4888
   0.4536    0.2189    0.5911    0.5473    0.8329
   0.5137   -0.0533    0.4111    0.2232    0.1028

 Columns 16 through 20
   0.7916    1.0849    0.2207    0.5245    0.1552
   0.5371    0.1152    0.2732    0.2567    0.5949
   0.3850    0.4233    0.2546    0.8712    0.4306
```

```
dd =
  -0.5000    8.0000   24.0000   -2.0000
  25.0000  -50.0000  210.0000         0


cinitial =
 Columns 1 through 5
   0.5389    0.1159    0.4699    0.7501    0.8551
   0.4371    0.3061    0.4188    0.7552    0.5496
   0.9183    0.5614    0.3575    0.3478    0.5220

 Columns 6 through 10
   0.4409    0.1959    0.4549    0.0912    0.0790
   0.2614    0.7416    0.9044    0.5476    0.1763
   0.4597    0.1071    0.6038    0.2359    0.9703

 Columns 11 through 15
   0.3318    0.3969    0.5011    0.6741    0.4390
   0.0249    0.5732    0.4064    0.1821    0.2168
   0.9892    0.1455    0.8094    0.7699    0.5015

 Columns 16 through 20
   0.3644    0.1053    0.1536    0.7967    0.9898
   0.1228    0.4237    0.1309    0.0669    0.9874
   0.7366    0.0988    0.6631    0.9087    0.6521


λinitial =
 Columns 1 through 5
   0.0522    0.3596    0.7912    0.0196    0.0944
   0.2187    0.3022    0.7395    0.9118    0.9155
   0.6915    0.7961    0.9498    0.0713    0.0520

 Columns 6 through 10
   0.6912    0.8497    0.6506    0.9057    0.6458
   0.8959    0.6622    0.1107    0.6442    0.6189
   0.2320    0.8362    0.0164    0.6206    0.6692

 Columns 11 through 15
   0.4579    0.5907    0.1419    0.7572    0.4888
   0.5231    0.5077    0.5931    0.6196    0.8329
   0.7234    0.2909    0.4706    0.4211    0.1027

 Columns 16 through 20
   0.8208    0.9586    0.3085    0.5766    0.3237
   0.6494    0.5601    0.3435    0.2485    0.6139
   0.7107    0.4831    0.5162    0.8895    0.6013
```

V =

0.7815	0.6420
0.1311	0.4381
-0.0518	0.0666
0.3834	0.0131
0.9654	0.3182
0.9410	0.0192
0.1908	0.5362
0.3619	0.7805
0.3309	-0.1305
0.0075	0.5197
0.7363	0.5835
0.7010	0.5960
0.7494	0.3222
0.5027	0.6261
0.5112	0.6621
0.1230	0.8822
-0.1300	0.7205
0.9575	0.4555
0.5990	0.0162
0.7022	0.6757

Vinitial =

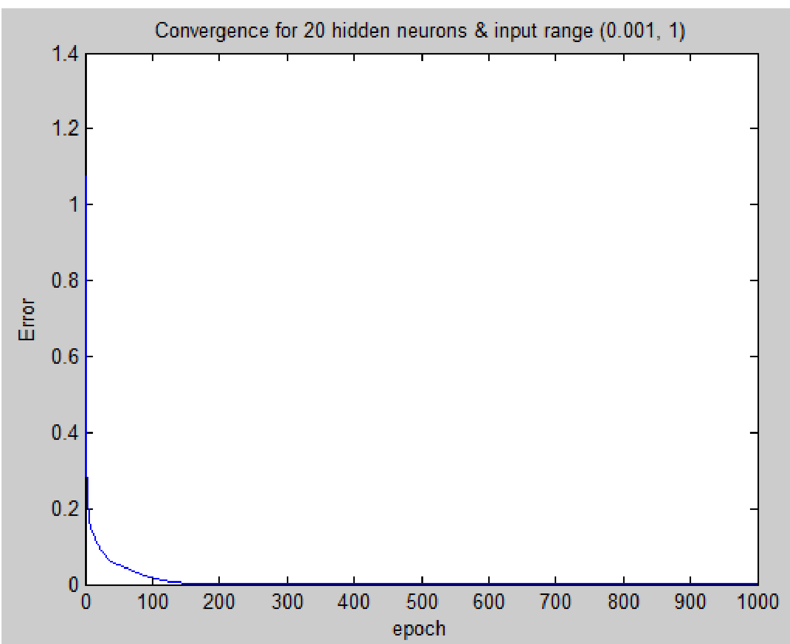
0.7978	0.6104
0.3772	0.1746
0.0599	0.1524
0.3834	0.0131
0.9654	0.3182
0.9556	0.0290
0.3972	0.2778
0.3619	0.7805
0.4457	0.0573
0.0480	0.5308
0.7668	0.6146
0.9324	0.7816
0.7549	0.3262
0.5220	0.6456
0.5112	0.6621
0.2313	0.9826
0.0859	0.4127
0.9853	0.4853
0.6119	0.0112
0.7686	0.7416

V0 =

0.0433
-0.0297

V0initial =

0.1000
0.1000



Appendix (code)

```
%scaling function
function[r] = scale(dmax, dmin, din, outmin, outmax)
    r = outmin + ((din - dmin)*(outmax-outmin)/(dmax-dmin));
end
%Edris Amin
clc;
clear all;
%define number of input, hidden, and output neurons
% nepoch = 1000; ux = 1; eta=0.1; nhid=2; %3 hidden H
nepoch = 1000; eta = 0.1; ux = 1;
nin=3; %3 input N
nhid=20; %2 hidden
nout=2; %2 output M

%training data
%define matrix for x input data:
%       1   2       3   ... number of tests
% x1    x11, x12, x13,  ... x1t
% x2    x21, x22, x23,  ... x2t
%
% number tests data = t= 5
t = 4;
xd=zeros(3, t); %training data
xin= zeros(3, t); %input data
xd= [6 8 10 12; -11 0 11 22; 0.1 1 10 100];
for i=1:t
%     for j=1:3
%         xin(j,i) = scale(100, -11, xd(j,i), 0, 1);
%         %       scale(dmax, dmin, din, outmin, outmax)
%     end
    xin(1,i) = scale(12, 6, xd(1,i), 0.001, 1);
        %       scale(dmax, dmin, din, outmin, outmax)
    xin(2,i) = scale(22, -11, xd(2,i), 0.001, 1);
        %       scale(dmax, dmin, din, outmin, outmax)
    xin(3,i) = scale(100, 0.1, xd(3,i), 0.001, 1);
        %       scale(dmax, dmin, din, outmin, outmax)
end


%desired output vector d=(M,1)
dd = zeros(nout, t); %given data
d = zeros(nout, t); %nueral data
%   1       2   3    4   ...num test
%y1 -0.5    8   24   -2
%y2 25      -50 210  0
dd= [-0.5 8 24 -2; 25 -50 210 0];
for i=1:t
    for j=1:nout
        d(j,i) = scale(210, -50, dd(j,i), 0, 1);
        %       scale(dmax, dmin, din, outmin, outmax)
    end
end

%matrix for c and l weights
%matrix of weights between input -- hidden c
%     c1      c2      c3
%x1   c11     c12     c13
%x2   c21     c22     c23
%x3   c31     c32     c23
%N+1  c01     c02     c03 !!! no hidden bias in this hw
%
% c = zeros(nin+1, nhid);
% l = zeros(nin+1, nhid);
c = zeros(nin, nhid);
l = zeros(nin, nhid);
%BP vector for dc = zeros(nhid, nin+1)
% dc = zeros(nin+1, nhid);
% dl = zeros(nin+1, nhid);
dc = zeros(nin, nhid);
```

```
dl = zeros(nin, nhid);
%     c1      c2      c3
%x1   dc11    dc12    dc13
%x2   dc21    dc22    dc23
%x3   dc31    dc32    dc23
%N+1  dc01    dc02    dc03 !!! no hidden bias in this hw
%

%derfine vector for gamma = g(size H)
%g is the internal value of the hidden neurons
g=zeros(nhid,1);

%define vector for Z = Z(size H)
%Z is output of hidden neuron
Z=zeros(nhid, 1);


%define vector for V = V(size H)
%    y1      y2
%h1  V11     V12
%h2  V21     V22
%
V=zeros(nhid, nout);
%V0(nout, 1) = output bias
V0 = zeros(nout, 1);

%BP vector for dV = zeros(nhid,1)
dV = zeros(nhid,nout);
%BP vector for output bias dV0 = zeros(nout,1)
dV0 = zeros(nout,1);

%define output vector y = zeros(M,t)
y=zeros(nout,t);


%randomize weights of links input -- hidden
%and randomize hidden layer bias (nin+1)
%matrix of weights between input -- hidden u
%     u1      u2      u3
%x1   u11     u12     u13
%x2   u21     u22     u23
%...    ...     ...
%N+1  u01     u02     u03
%
for i = 1:(nhid);
    for j = 1:1:(nin);
        c(j, i)= rand*ux;
        l(j, i)= rand*ux;
    end
end
%u = [0.1985 0.1985 0.1985; 0.1979 0.1979 0.1979; 0.199 0.199 0.199];
%u = [0.1985; 0.1985; 0.1985];
c1 = c;
l1 = l;

%Randomize V = zeros(H, M)
for j= 1:1:nhid;
    for i=1:nout;
        V(j,i) = rand;
    end
end
% V = [0.997; 0.996; 0.995];
V1 = V;
%randomize output bias V0 = zeros(nout,1)
for j = 1:1:nout
    V0(j, 1) = 0.1;
end
% V0=0.99;
V01 = V0;

%create Error vector E = zeros(nout, #tests
```

```
E=zeros(nout, t);
%dE = dE/dY = zeros(#out, training size)
dE = zeros(nout, t);

Eavg = zeros(nepoch, 1);
%********Epochs
for epoch = 1:nepoch

    for test = 1:t; %sample by sample to sample number t
        %+++++++++++ Feed-forward
        %calculate gamma vector
        %per testNUMBER test
        %per x(i, test)
        %for test = 1:1:t --> run through all test cases
        for j = 1:nhid
            g(j,1) = 0;
            for i = 1:nin;
                g(j,1) = g(j,1) + ((xin(i,test)-c(i,j))/(l(i,j)))^2;
                %0 + ((x(1,test)-c(1,j))/(l(1,j)))^2 +
                %((x(2,test)-c(2,j))/(l(2,j)))^2 +
                %((x(3,test)-c(3,j))/(l(3,j)))^2
            end
            g(j,1) = sqrt(g(j,1));
        end

        %calculate Z vector Z(H,1)
        for j = 1:nhid;
            Z(j,1)=exp(-(g(j,1))^2);
        end

        %calculate output vector y(j,test) = V0(j,1)+Z(j)*V(j)
        for i = 1:nout
            y(i,test) = V0(i,1); %init with bias

            for j = 1:nhid
                y(i,test) = y(i,test)+Z(j,1)*V(j,i);
            end
        end

        %+++++++++++END feed-forward
        %Error Calculations 'E' = zeros(1, test)
        %columns = number of training data
        %e1=0.5*((y-d(1,test))^2);
        %E(1, test)=0.5*((y(1,test)-d(1,test))^2);
        %dE/dY = (y-d);
        for k = 1:nout
            dE(k, test) = (y(k,test)-d(k,test));
        end

        %====== Back Propagation
        %find dV and dV0
        for k = 1: nout
            for j=1:nhid
                dV(j,k) = dV(j,k)+ dE(k, test) * Z(j);
                %dE/dV=        dE/dy        * dy/dV
            end
        end
        for k = 1:nout
            dV0(k) = dE(k, test);
        end

        %find dc = zeros(nin, nhid) = dE/dc;
        for k = 1:nout
            for i=1:nhid
                for j=1:nin
                    dc(j,i) = dc(j,i) + dE(k, test)*V(i,k)*(-2*g(i)*Z(i))*((c(j,i)-
xin(j,test))/((l(j,i))^2*g(i)));
                    %dE/dc =            dE/dy *    dy/dZ * (dZ/dg      ) *  dg/dc;
                end
            end
        end
```

```
        %find dl = zeros(nin, nhid) = dE/dl;
        for k = 1:nout
            for i=1:nhid
                for j=1:nin
                    dl(j,i) = dl(j,i)+ dE(k, test)*V(i,k)*(-2*g(i)*Z(i))*(-((c(j,i)-
xin(j,test))^2/((l(j,i))^3*g(i))));
                    %dE/dl =        dE/dy    *      dy/dZ *(dZ/dg       )* dg/dl
                end
            end
        end
%      c=c-dc.*eta;
%      l=l-dl.*eta;
%      V=V-eta.*dV;
%      V0=V0-eta.*(dE(nout,test)).*dV0;
%      dc = zeros(nin, nhid);
%      dl = zeros(nin, nhid);
%      dV = zeros(nhid,nout);
%      dV0 = zeros(nout,1);
        for i = 1:t
           for s =1:nout
               yo(s,i) = scale(1, 0.01, y(s,i), -50, 210);
               %         scale(dmax, dmin, din, outmin, outmax)
           end
        end
    end %end run of all samples
    %update weights
    %w=w(-eta*dE/dW); eta = 0.1
    %eta = 0.1;

    c=c-dc.*eta;
    l=l-dl.*eta;
    V=V-eta.*dV;
    V0=V0-eta.*dV0;
%     rezero all C and L's for additive averaging
    dc = zeros(nin, nhid);
    dl = zeros(nin, nhid);
    dV = zeros(nhid,nout);
    dV0 = zeros(nout,1);
%     Escaled = 0.5*(y-d).^2;
    Escaled=abs(y-d);
    E = sum(Escaled);
    Eavg(epoch)= sum(E)/(2*numel(E));
end

for i = 1:t
   for s =1:nout
       yo(s,i) = scale(1, 0, y(s,i), -50, 210);
       %         scale(dmax, dmin, din, outmin, outmax)
   end
end
yo
dd
Eavg(nepoch)
Escaled
Eall = (yo-dd)
epoch
plot(1:1:nepoch, Eavg);
xlabel('epoch');
ylabel('Error');
c
c1
l
l1
V
V1
V0
V01
```